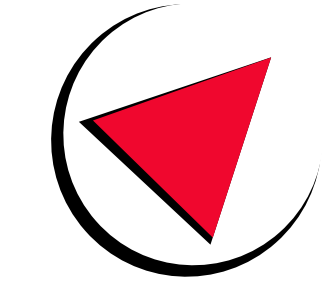
A Genetic Algorithm to Schedule Workflow Collections on a SOA-Grid with Communication Costs

Jean Nicod, Laurent Philippe, Lamiel Toch



L I F C

LABORATOIRE D'INFORMATIQUE DE L'UNIVERSITE DE FRANCHE-COMTE

EA 4269

A Genetic Algorithm to Schedule Workflow Collections on a SOA-Grid with Communication Costs

Jean-Marc Nicod and Laurent Philippe and Lamiel Toch

Rapport de Recherche no RR 2011–01

THÈME 1 – 2011

A Genetic Algorithm to Schedule Workflow Collections on a SOA-Grid with Communication Costs

Jean-Marc Nicod and Laurent Philippe and Lamiel Toch

Thème 1

CARTOON

2011

Abstract: In this paper we study the problem of scheduling a collection of workflows, identical or not, on a SOA grid. A workflow (job) is represented by a directed acyclic graph (DAG) with typed tasks. All of the grid hosts are able to process a set of task types with unrelated processing costs and are able to transmit files through communication links for which the communication times are not negligible. The goal is to minimize the maximum completion time (makespan) of the workflows. To solve this problem we propose a genetic approach. The contributions of this paper are both the design of a Genetic Algorithm taking the communication costs into account and the performance analysis.

Key-words: Batch scheduling, grid computing, heterogeneous platform, genetic algorithm

Laboratoire d'Informatique de l'Université de Franche-Comté,
UFR Sciences et Techniques,
16, route de Gray, 25030 Besançon Cedex (France)
Téléphone : +33 (0)3 81 66 64 55 — Télécopie : +33 (0)3 81 66 64 50

Algorithme Génétique pour Ordonnancer un Ensemble de Workflow sur une Grille Orientée Service avec des Coûts de Communications

Résumé : Dans ce rapport de recherche nous considérons un problème pour ordonnancer un ensemble de workflows, identiques ou non, sur une grille fondée sur une architecture orientée service. Un workflow (job) est représenté par un graph acyclique orienté avec un des tâches typées. Chacune des machines de la grille savent exécuter un ensemble de type de tâches prédéfinies avec des coûts indépendants les uns des autres. L'objectif est de minimiser la date de fin de la derniére tâche (le makespan) de l'ensemble de workflows. Pour résoudre ce probléme nous proposons une approche génétique. Nous apportons une contributions à la fois dans la conception d'un algorithme génétique prenant les coûts de communication en compte et dans les analyses des performances.

Mots-clés : Ordonnancement de lots, grille de calcul, plateforme hétérogène, algorithme génétique

Laboratoire d'Informatique de l'Université de Franche-Comté,
UFR Sciences et Techniques,
16, route de Gray, 25030 Besançon Cedex (France)
Téléphone : +33 (0)3 81 66 64 55 — Télécopie : +33 (0)3 81 66 64 50

1 Introduction

Nowadays, to go further in their research, scientists often need to connect several applications together. That is why for few years, workflow systems have been designed to provide tools that support their multi-application simulations. In the e-Science [22] many fields use workflow applications: medical image processing [14], geosciences [11], astronomy [20].

A workflow is a set of applications which are connected to each other by precedence constraints. An input data set enters the workflow, it is processed by an application which computes an output data set. This result data set is sent to the next application as defined by the structure of the workflow. Generally a workflow has the structure of a DAG (Direct Acyclic Graph): a graph whose nodes are the tasks and whose edges are the precedence constraints.

When the size of the data entering the workflow increases the processing time may become very long and it becomes mandatory to use larger computing resources as grids. Because of their heterogeneity – the platforms have hosts with different calculation and memory capacities – they are difficult to use for non computer scientists but several research projects have already tackled this problem. The Pegasus framework [12] proposes a convenient way for scientists to compute their workflows onto heterogeneous platforms without learning distributed programming concepts. Other tools, like DIET [5] or NINF-G [21], provide a SOA-Grid (Service Oriented Architecture) that facilitates user accesses to remotely accessible computing applications and make the execution of workflows on a heterogeneous platform easier. When the number of workflows to be executed in parallel is huge, it is however mandatory to efficiently map them onto the resources of a heterogeneous platform.

Minimizing the execution time of a workflow or a set of workflows onto a heterogeneous platform is an optimization problem that is known to be NP-Hard. We can just rely on heuristics to find a solution as good as possible. The quality of the heuristic solution is based on the performance obtained in the scheduling of different workflows. Genetic Algorithms (GA) are known to give good results in several optimization domains. GA have already been used in the scheduling of workflows [8, 15]. With these algorithms it is however not possible to give an approximation and a static study of the algorithm may be worth it. The performance evaluation must be done by simulating the algorithm behavior in a large number of cases.

The general problem we deal with is to schedule a set of workflows (jobs) onto a SOA-grid. Each task of a workflow has a type that corresponds to a service type or to an application type in the SOA-Grid. We consider the two cases: (1) the general case when the structure of each workflow

differs from one another; (2) the particular case when the structure of the workflow is the same for all the jobs. In this paper we focus on researches that we carried out on the design of a Genetic Algorithm and we assess its performance to schedule a set of workflows. The contributions of this paper are both the design of a Genetic Algorithm taking the communication costs into account and the performance analysis. In the general case (1) we compare the performance obtained by a GA approach to the performance obtained by a list-based scheduling algorithm. In the particular case (2) of the scheduling of a collection of identical workflows which structure is limited to intrees, we compare our results to a lower bound and we show that our Genetic Algorithm approach allows us to get a schedule with good performance.

The paper is organized as follows. In section 2 we detail the framework which defines both the grid and the workflow models. Section 3 is devoted to an overview of the related work. We present in section 4 a Genetic Algorithm that takes communication costs into account to deal with our problem. Section 6 introduces the simulation setup, the results of the simulations and their analysis for workflows of different shape. Section 7 presents the simulations for workflows with identical structure. Finally, we conclude in section 8.

2 Framework

Scheduling a set of workflows on a SOA-grid amounts to schedule each task of each workflow onto a grid host that provides the application corresponding to its task type. In this section we outline the applicative framework, the targeted platform and the communication model.

2.1 Applicative framework

In our study, the problem that we have to solve deals with the scheduling of a collection $\mathcal{B} = \{\mathcal{J}^j, 1 \leq j \leq N\}$ of N workflows. Each workflow $\mathcal{J}^j$ is represented by the graph (DAG) $\mathcal{J}^j = (\mathcal{T}^j, \mathcal{D}^j)$ where $\mathcal{T}^j = \{T_1^j, \ldots, T_{n_j}^j\}$ is the set of its tasks and $\mathcal{D}^j$ represents the precedence constraints between the tasks of the workflow. Let $\mathcal{T} = \cup_{j=1}^{N} \mathcal{T}^j = \{T_{i_j}^j, 1 \leq i_j \leq n_j \text{ and } 1 \leq j \leq N\}$ be the set of tasks to schedule.

Figure 1 gives an example of a workflow $\mathcal{J}^1$ that consists of three tasks T_1^1, T_2^1 and T_3^1 which types are respectively A, D and B. Each edge of $\mathcal{J}^1$ indicates a precedence constraint and its associated value is the amount of data that has to be exchanged. These precedence constraints actually represent file transfers between applications. These transfers, that may be costly,

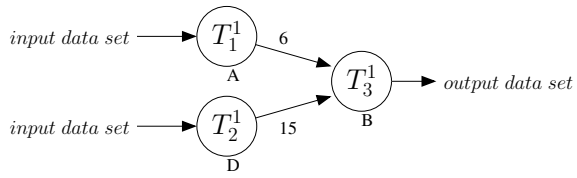


Figure 1: Example of a workflow $\mathcal{J}^1$

implies that communication costs must be taken into account in the schedule. As we mention above we distinguish the two following cases: scheduling identical workflows or scheduling different workflows.

2.2 Targeted platform

The targeted platform is a heterogeneous platform of n machines modeled by an undirected graph $\mathcal{PF} = (\mathcal{P}, \mathcal{L})$ where $\mathcal{P} = \{p_1, \ldots, p_n\}$ is the set of vertices which represents the machines and where $\mathcal{L}$ is the set of bidirectional communication links between machines ($\forall\ 1 \leq i, j \leq n$ and $i \neq j$ then $(p_i, p_j) \in \mathcal{L}$).

Let τ be the set of all the task types available onto the platform. Each machine p_i is able to perform a subset of τ. If the type $t \in \tau$ is available on the machine p_i, $w(t, p_i)$ is the time to perform a task of type t on p_i. Moreover, each link (p_i, p_j) has a bandwidth $bw(p_i, p_j)$ which is the number of data per time unit that can be transferred through that link.

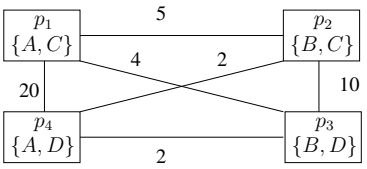


Figure 2: An example of a platform

Figure 2 gives an example of a platform where the letters A, B, C and D represent the task types that can be performed onto the nodes. The number above the link is the bandwidth.

Figure 3 gives an example of the capabilities of each machine to perform each task type. The values correspond to the number of time units necessary to carry out the corresponding task type onto a given processor. When the value is infinity, the machine cannot perform the task type. For instance, processor p_1 can perform both the functions A and C but cannot perform neither B nor D.

		p_1	p_2	p_3	p_4
Type	A	20	∞	∞	15
	B	∞	10	5	∞
	C	10	10	∞	∞
	D	∞	∞	10	10

Figure 3: Execution platform performance.

2.3 Communication model

In our study processors are interconnected by communication links in a point-to-point fashion to model a computation grid. In the literature [1, 6], several communication models exist such as the one-port model or the all-port model. We choose to use the one-port model because of its ease of modeling and of implementation while still being realistic. In this model only one data can be transmitted at the same time over a communication link and a node can do at most one reception and one transmission at the same time.

3 Related work

Scheduling problems have extensively been studied in the literature and minimizing the makespan (maximum completion time) of a DAG execution on a distributed homogeneous platform with limited resources, i.e., a limited number of processors, is known to be a NP-Complete problem [18]. As a consequence, the problem considered here is also NP-Complete since the heterogeneity of the resources makes the problem more complex. So there is no direct method that can find an optimal solution in polynomial time and the schedule decisions must rely on heuristic solutions.

Classical solution scheduling algorithms often rely on list-based scheduling algorithms. They use heuristics such as Heterogeneous Earliest Finish Time (HEFT) [23] or Critical Path [16]. In a homogeneous context, [17] gives a survey on the DAG scheduling topic. Moreover the more complete study [13] evaluates eleven heuristics, such as Min-min, Max-min or Sufferage, three heuristics based on the Minimum Completion Time (MCT). These results obtained onto homogeneous platforms give hints to schedule DAGs onto heterogeneous platforms but have not been validated for Heterogeneous Computing.

Makespan oriented strategies to schedule workflows onto the grid are presented in [19], with the description of real life medical applications, or in [4] for Ocean-Atmosphere modeling. In [25], the problem of scheduling a set of

different DAGs is studied. These approaches compute an off-line schedule considering the whole set of tasks. But, when the number of tasks scales up, the computation time becomes too long because of the complexity of the algorithm.

M. Daoud and N. Kharma present in [8] a makespan oriented genetic based algorithm (GA) for tasks scheduling. It has been designed for scheduling tasks of one workflow onto a heterogeneous platform but without taking communication costs into account. C. Goh et al. use in [15] the same genome coding but communications are not handled too.

Other studies tackle workflow scheduling onto heterogeneous platforms but with other objective functions than the makespan. The results presented in [2] use Timed Petri Nets and pipelines to optimize the throughput of replicated workflows onto heterogeneous platforms. The steady-state technique presented in [1] provides an optimal algorithm to schedule a set of identical workflows also for the throughput objective function. Other optimization criteria are studied in the literature, such as the latency [24], or even performance-related criteria together with energy consumption [3].

4 GA with communication costs

For the execution we use the same genome representation as in [8, 15], i.e., a chromosome is a two-dimension table with one row per node where the tasks assigned to the node are recorded. Some improvements to take SOA-Grids and collections into account, presented in [10], are added. As these GA coding does not however take communication costs into account, we study their integration in this section.

4.1 Communication integration

The main problem we faced is the step of the genetic algorithm where we should integrate the communications.

At first we have worked on inserting communications into the chromosomes, as for executions. We have introduced the notion of "*communication task*", between each couple of dependent tasks in the task graph. Then we map them onto the communication links. When integrating "*communication tasks*" in the chromosomes, there are however issues regarding the crossover procedure. Let us imagine two schedules for one application onto one platform. In each schedule a task is allocated to one node, and its output (a communication) is allocated to a route composed of connected communication links. Now, if we make a crossover with these two schedules, and if we

apply the crossover technique on communication tasks with their allocated links, we potentially generate a lot of inconsistent communication routes and we thus produce numerous non feasible schedules. This makes a huge waste in term of computation time.

A more convenient way to introduce communications is at the fitness evaluation step of the genetic algorithm. Indeed the allocations of the communication tasks depend on the allocations of the computation tasks since the possible routes between two nodes are limited. Choosing a good mapping for the computation tasks intuitively involves finding a valid allocation for the communication tasks. So we propose to use the chromosomes only for mapping computation tasks onto the nodes, and in a second step to look for a valid route to send the data according to this mapping and the precedence constraints between the tasks. Then the fitness of each individual can be evaluated by taking both the computations and the communications into account, each individual being valid (see algorithm 1).

4.2 Impact of the communication model

Choosing the network links for each communication is complex. Always using the shortest, or fastest, path may lead to a high contention so that we need to take the network load into account. We assess here four communication policies.

Figures 4a and 4b show the efficiencies of the GA different communication integrations. This notion of efficiency will be introduced in section 7.1, however, it is necessary to know only that the higher the curve is, the higher the performance of the algorithm is.

- "GA-without-comm" : an arbitrary static route is given for each couple of computation nodes. Chromosome fitness are evaluated without regarding the communication cost.

- "GA 1-port-comm-static-route" : the same the previous policy but with regarding communication costs in the fitness evaluation using the 1-port model.

- "GA 1-route-Bellman-Ford" : the static route for each couple of computation nodes is the fastest one by using the Bellman-Ford algorithm.

- "GA 3-routes-Bellman-Ford" : for each couple of computation nodes, during the evaluation fitness step, the 3 first best routes are tested.

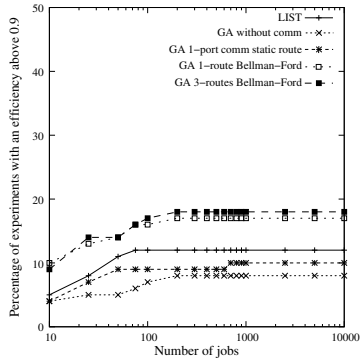
Algorithm 1: Scheduling tasks according the chromosome ch

Input : $\mathcal{B}$: the N DAGs to schedule
$a(i,j)$: $p_{a(i,j)}$ is the machine on which T_i^j is assigned
$\mathcal{PF} = (\mathcal{P}, \mathcal{L})$: the targeted platform
$\mathcal{R}(p_k, p_i) = \{(p_j, p_{j'}) \in \mathcal{L}\}$: a route from p_k to p_i
$t(i,j)$: the type of the task T_i^j
$w(t(i,j), u)$: the time to perform T_i^j onto p_u

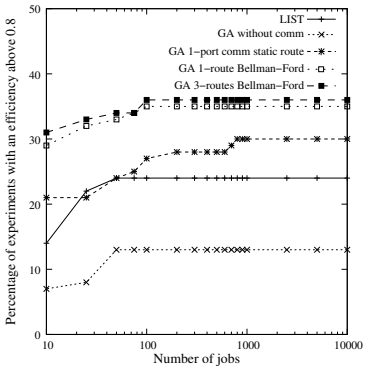
Output: $f(ch)$: the fitness associated to the chromosome ch

Data: $\mathcal{T} = \{T_{i_j}^j, 1 \leq i_j \leq n_j \text{ and } 1 \leq j \leq N\}$: the set of tasks to schedule
$\mathcal{T}_{ToSched}$: the set of remaining tasks to schedule
$C(T_i^j)$: the completion time of T_i^j
$\sigma(T_i^j)$: the time to start T_i^j onto $p_{a(i,j)}$
$\delta(p_u)$: the next time where p_u is idle
$F_{k,i}^j$: the file to send between T_k^j and T_i^j when $(T_k^j, T_i^j) \in \mathcal{D}^j$
$CF(F_{k,i}^j)$: the time to send $F_{k,i}^j$ along the route $\mathcal{R}(p_{a(k,j)}, p_{a(i,j)})$

$\mathcal{T}_{ToSched} \leftarrow \mathcal{T}$
while $\mathcal{T}_{ToSched} \neq \emptyset$ **do**
  choose a free task $T_i^j \in \mathcal{T}_{ToSched}$ (EFT heuristic) $\mathcal{T}_{pred} \leftarrow \{T_k^j | (T_k^j, T_i^j) \in \mathcal{D}^j\}$
  $\sigma(T_i^j) \leftarrow 0$
  foreach *task* $T_k^j \in \mathcal{T}_{pred}$ **do**
    $\sigma(T_i^j) \leftarrow \max(\sigma(T_i^j), C(T_k^j) + CF(F_{k,i}^j))$
  $\sigma(T_i^j) \leftarrow \max(\delta(p_{a(i,j)}), \sigma(T_i^j))$
  $C(T_i^j) \leftarrow \sigma(T_i^j) + w(t(i,j), a(i,j))$
  $\delta(p_{a(i,j)}) \leftarrow C(T_i^j)$
  The task T_i^j is scheduled at the time $\sigma(T_i^j)$ onto $p_{a(i,j)}$
  $\mathcal{T}_{ToSched} \leftarrow \mathcal{T}_{ToSched} \setminus \{T_i^j\}$
return $f(ch) = C_{max} = max_{T_i^j \in \mathcal{T}}(C(T_i^k))$



(a)



(b)

Figure 4: Communication integrations with an efficiency above 0.9 and 0.8, $CCR \approx 10$

Comparing this preliminary work to performance of a list-based scheduling (LIST) allows us to see that taking communication into account leads to good performance.

5 Simulation setup

To assess the performance of the scheduling algorithm we need to implement it on a heterogeneous platform. The context is indeed too complex to be studied with a formal approach and, to get realistic results, the platforms used must integrate the network contention. On the other hand, the implementation on a computation grid can however not give reproducible results as the experimental conditions, as the network load, may change. So, we use a grid simulator to evaluate the performance of the Genetic Algorithm. The simulator is implemented using SimGrid and its MSG API [7].

All the simulations have been made using batch sizes from 1 to 10 000. The platforms and the applications are randomly generated with a uniform distribution. Platforms have between 4 and 10 nodes and are strongly connected. Applications have between 4 and 12 tasks. For the case where workflows are different from each other, presented in 6, 200 platforms and 10 000 DAGs are randomly generated. For each couple (platform, batch size) different applications are randomly chosen among the 10 000. So we generate 1 900 simulations of platform/application scenarios. In the case where workflows are identical, we use 10 platforms and 10 applications, so 100 scenarios for each batch size.

We define the "*computation to communication ratio*" (CCR) of a simulation as the average computation time divided by the average communication time. To assess the impact of the communications on the algorithm performance, we run simulations with different CCR, from 1/1 000, i.e., communication time is predominant, to 1 000, i.e., communication time is negligible. The speed of the nodes are unrelated. We also assess the impact of the platform heterogeneity on the performance: execution and communication times fluctuate respectively in a range from 1 to 10 and 1 to 4.

For the GA, the population is set to 200 individuals and a generation is set to 100 iterations.

6 Results with general DAGs

As no optimal value can be computed in a polynomial time for the global execution time of a collection of workflows, we must compare the performance

of the GA to another algorithm. So the metric that we use is the makespan improvement relative to a standard list-based scheduling algorithm (LIST).

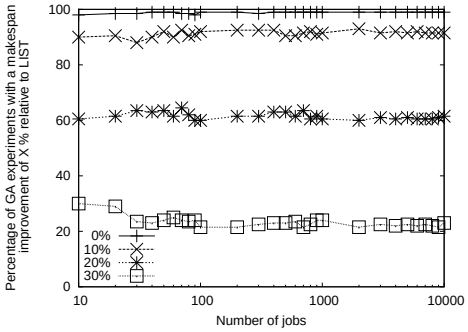


Figure 5: Improvement for batches of different DAGs, fully heterogeneous platforms, $CCR \approx 1$

Figure 5 shows the improvement of the makespan when scheduling a set of different workflows on a heterogeneous platform. Each curve shows the percentage of GA experiences whose makespan is improved respectively by more than 0%, 10%, 20% and 30% when compared to the use of the list-based scheduling algorithm (LIST). We notice that more than 80% of the GA experiments have a makespan improved by more than 10%. The 0%-curve shows that GA gives an improvement in about 100% of cases.

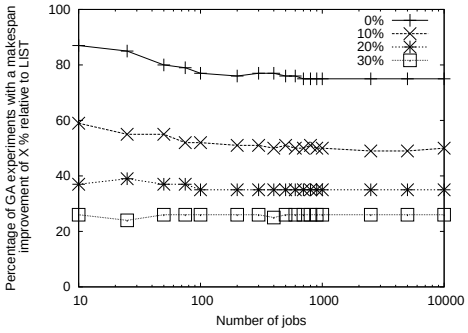


Figure 6: Improvement for batches of identical general DAGs, fully heterogeneous platforms, $CCR \approx 1$

Figure 6 shows the improvement of the makespan when scheduling a set of identical workflows. We can notice that GA gives less important im-

provements relative to LIST in this case. For each 0, 10, 20 and 30 %-curve, GA improvements decrease by about 10% compared to experiments obtained when scheduling different workflows. We can also note that 10% to 20% of the GA schedules are less efficient than the LIST schedules in this case. GA improvements remain however high.

In general, more than 70% of the GA experiments give a makespan with an improvement greater than 10% for almost any size of sets of workflows. The diversity of the workflows promotes the GA algorithm: as GA tries randomly a lot of combinations it can benefit from different workflows while LIST which is directed by greedy choices, cannot.

7 Results for a set of identical intrees

Comparing two algorithms gives relative information but no absolute information on the quality of the algorithm. For the particular case of batches of intrees it is possible to compute an optimal throughput by using a steady-state algorithm [1]. In this section we assess how far the GA is from the optimal, so its absolute efficiency, and we compare it to the list scheduling and to the a practical makespan oriented implementation of this algorithm [9].

Note that this study is limited to the particular case where the applications are intrees.

7.1 Efficiency

Using linear programming the optimal throughput can be computed thanks to the steady-state algorithm. This optimal throughput is used to compute a lower bound for the optimal makespan $makespan_o$ which the number N of jobs to process divided by the steady-state throughput (ρ). To evaluate the GA performance, we introduce the notion of efficiency as the ratio between this lower bound $makespan_o$ over the makespan of the schedule given by GA $makespan_r$. That is to say if a schedule gives $makespan_r = makespan_o$ then $efficiency = 1.0$ and it means that this schedule is optimal. The nearer efficiency is from 1.0, the more efficient the scheduling algorithm is.

$$efficiency = \frac{makespan_o}{makespan_r} \qquad \text{with} \qquad makespan_o \geq \frac{N}{\rho}$$

$$\text{So, } efficiency \geq \frac{N}{\rho \times makespan_r}$$

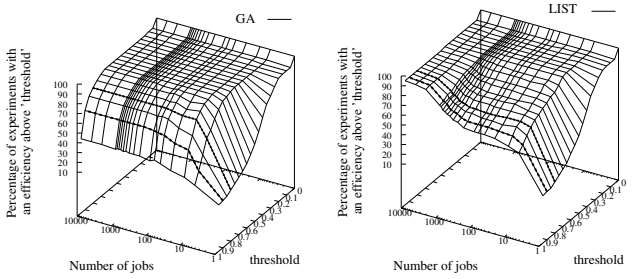
Since we run 100 simulations, for each of the 19 sizes of job sets, we cannot get a simple scalar efficiency value for the overall experiments. A mean value

would indeed be not meaningful as the longer simulations will weight more than the sorter ones in this metric. So we introduce an efficiency threshold t for the different sizes of batches $SIZES$ and we compute the percentage of experiments that gives an efficiency greater than t for each size of batches.

In the following we present the distribution of the results depending on this percentage on 3D curves. Two lines are thickened on the surfaces to highlight the efficiency curves for threshold values 0.8 and 0.9.

7.2 Fully heterogeneous platforms

In this section we compare the performance of GA, LIST and steady-state to schedule intrees on heterogeneous platforms.

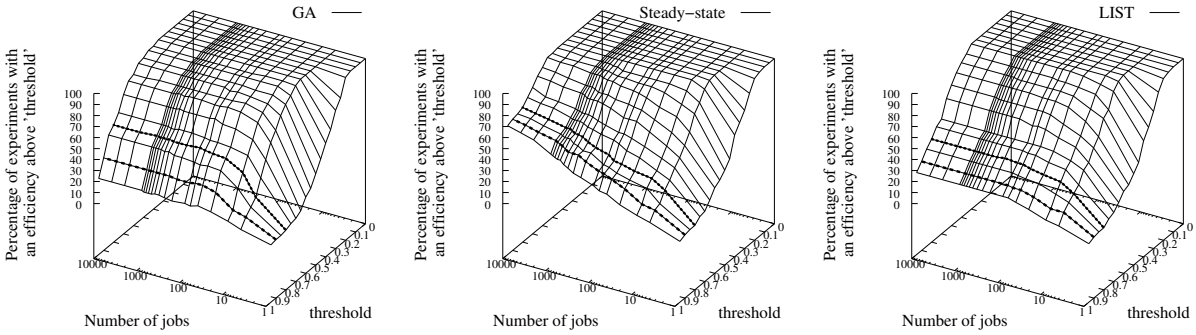


(a) GA algorithm (b) LIST algorithm

Figure 7: $CCR \approx 1000$, fully heterogeneous platforms

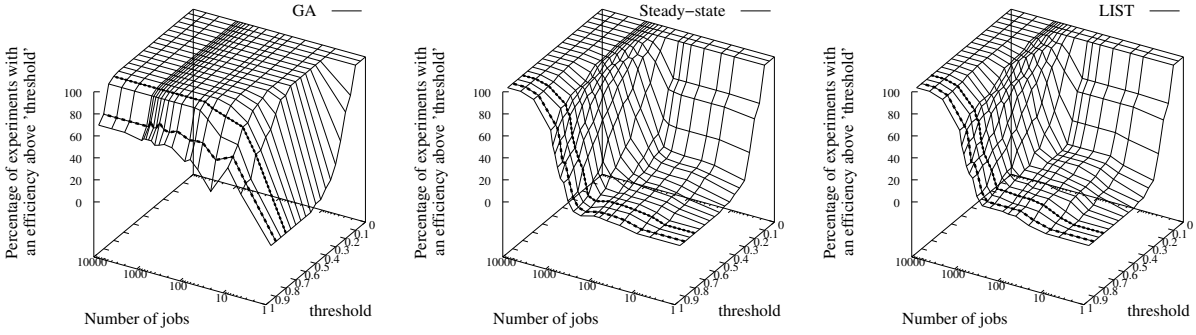
Figures 7a and 7b show the distribution of the percentage of experiments which give schedules greater than a threshold and for a batch size. The used platforms are fully heterogeneous and $CCR \approx 1\,000$. For both the algorithms the efficiency increases with the number of jobs. We can notice that for batch sizes greater than 100 almost 100% of GA experiments reach an efficiency greater than 0.8. There is an 100% horizontal line for GA experiments for threshold 0.8 and batch sizes from 100 to 10 000, while for LIST there is a rising curve. It means that GA efficiency increases faster than LIST when batch sizes are large.

Figures 8a and 8c show that for batch sizes greater than 100 more than 70% of GA experiments have an efficiency greater than 0.8, while LIST reaches hardly 60%. However for the threshold 0.9 the two experiments have almost the same behavior. On the other hand, the steady-state algorithm gives better results with almost 90% of the simulations that reach an efficiency of 0.8. The GA is however not so far from this curve.



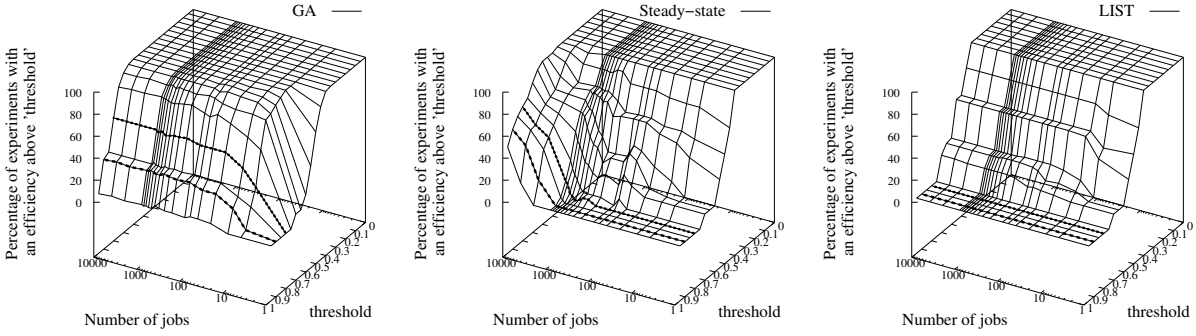
(a) GA algorithm (b) Steady-State algorithm (c) LIST algorithm

Figure 8: Simulation with fully heterogeneous platforms, $CCR \approx 1$



(a) GA algorithm (b) Steady-State algorithm (c) LIST algorithm

Figure 9: Simulation with fully homogeneous platforms, $CCR \approx 1000$



(a) GA algorithm (b) Steady-State algorithm (c) LIST algorithm

Figure 10: Simulation with fully homogeneous platforms, $CCR \approx 1$

7.3 Fully homogeneous platforms

In this section we assess the performance of GA for scheduling intrees on homogeneous platforms.

7.3.1 $CCR \approx 1\,000$

On the figures 9a, 9b and 9c we see the percentage of experiments which give schedules with efficiencies greater than 0.9 and 0.8 when the platforms have heterogeneous nodes and heterogeneous communication links. For the threshold 0.9, GA is even better than both the Steady-State and the LIST algorithms for batch sizes less than 1 000. For batch sizes greater than 1 000, GA is overtaken. However if threshold 0.8 is taken, GA shows the highest curve.

7.3.2 $CCR \approx 1$

With the figures 10a, 10b and 10c we see the percentage of experiments which give schedules with efficiencies greater than 0.9 and 0.8 when the platforms have homogeneous nodes and homogeneous communication links. For the threshold 0.9, GA is even better than both the Steady-State and the LIST algorithms for batch sizes less than 2 500. For batch sizes greater than 2 500, GA is overtaken. However if threshold 0.8 is taken, GA shows the highest curve up to 5 000 jobs. Figure 10a shows that LIST has very poor results even for several thresholds.

7.4 Computation times

The figure 11 shows the time spent by the three algorithms to find a schedule for platforms with heterogeneous bandwidths and whose computation costs are 10 times greater than communication costs. The GA and LIST are very time consuming while the computation time spent by the Steady-State algorithm is very low. Nevertheless GA has the best performance in most seen cases for batch sizes less than 1 000. For these sizes the GA computation times is less than 20 minutes only.

The simulations have been run on a cluster with 64 nodes, each node being a 2.8 GHz quad-core Intel Xeon bi-processor. Each simulation runs on one core.

8 Conclusion and future work

In this paper, we propose a genetic algorithm that solves the problem of scheduling a collection of workflows on a set of heterogeneous nodes inter-

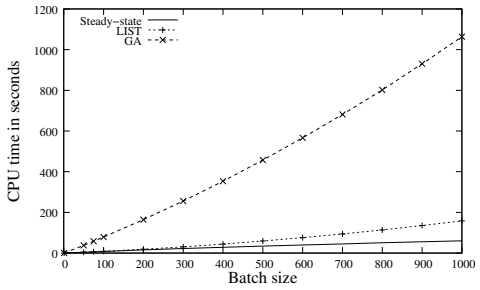


Figure 11: Average Computation time

connected by heterogeneous communication links. This GA takes the communication costs into account. We show that for a collection of different workflows GA obtains better execution performance than a classical LIST algorithm. For the case where the DAGs are identical intrees we are able to compare the GA results to an optimal lower bound and to show that its results tend towards the optimality for more than 1 000 jobs. The obtained results are moreover comparable to a practical implementation of the Steady-State algorithm. The main idea we keep in mind is that scheduling with GA by taking communications into account is difficult, as for implementing the practical Steady-State solution.

For future work, other communication models should be implemented in the simulator, like the multi-port models and other genetic representations could be explored.

Acknowledgment

The huge amount of simulations have been run on the cluster of the *Mésocentre de Calcul de Franche-Comté* in Besançon, France.

References

[1] O. Beaumont, A. Legrand, L. Marchal, and Y. Robert. Assessing the impact and limits of steady-state scheduling for mixed task and data parallelism on heterogeneous platforms. In *HeteroPar'2004*, pages 296–302, 2004.

[2] A. Benoit, B. Gaujal, M. Gallet, and Y. Robert. Computing the throughput of replicated workflows on heterogeneous platforms. In *Proceedings of ICPP'09*, pages 204–211, 2009.

[3] A. Benoit, P. Renaud-Goud, and Y. Robert. Performance and energy optimization of concurrent pipelined applications. In *IPDPS'2010, the 24th IEEE International Parallel and Distributed Processing Symposium*. IEEE Computer Society Press, 2010.

[4] Y. Caniou, E. Caron, G. Charrier, A. Chis, F. Desprez, and Maisonnave E. Ocean-Atmosphere Modelization over the Grid. In *ICPP'08*, pages 206–213, 2008.

[5] E. Caron and F. Desprez. Diet: A scalable toolbox to build network enabled servers on the grid. *International Journal of High Performance Computing Applications*, 20(3):335–352, 2006.

[6] H. Casanova. Modeling large-scale platforms for the analysis and the simulation of scheduling strategies. In *IEEE APDCM'04*. IEEE Computer Society, 2004.

[7] H. Casanova, A. Legrand, and M. Quinson. Simgrid: A generic framework for large-scale distributed experiments. In *Proceedings of UKSIM '08*, pages 126–131, 2008.

[8] M. Daoud and N. Kharma. GATS 1.0: A Novel GA-based Scheduling Algorithm for Task Scheduling on Heterogeneous Processor Nets. In *Genetic And Evolutionary Computation Conference*, 2005.

[9] Sékou Diakité, Loris Marchal, Jean-Marc Nicod, and Laurent Philippe. Steady-state for batches of identical task graphs. In *Euro-Par'09*, volume 5704 of *LNCS*, pages 203–215, Delft University of Technology, Delft, the Netherlands, August 2009.

[10] Sékou Diakité, Jean-Marc Nicod, and Laurent Philippe. Comparison of batch scheduling for identical multi-tasks jobs on heterogeneous platforms. In *PDP 2008*, pages 374–378, Toulouse, France, 2008.

[11] E. Deelman et al. Managing Large-Scale Workflow Execution from Resource Provisioning to Provenance Tracking: The CyberShake Example. In *Proceedings 2nd IEEE Int. Conf. on e-Science and Grid Computing*, 2006.

[12] Ewa Deelman et al. Pegasus: a framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming Journal*, 13:219–237, 2005.

[13] T.-D. Braun et al. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *J. Parallel Distrib. Comput.*, 61:810–837, June 2001.

[14] V.-S. Kumar et al. Large image correction and warping in a cluster environment. In *Proceedings SC'06 Conference*, page 79, Tampa, USA, 2006. ACM.

[15] C. K. Goh, E. J. Teoh, and K. C. Tan. A hybrid evolutionary approach for heterogeneous multiprocessor scheduling. *Soft Comput.*, 13:833–846, March 2009.

[16] Y. Kwok and I. Ahmad. Dynamic critical-path scheduling: An effective technique for allocating task graphs to multi-processors. In *IEEE Trans. on Parallel and Distributed Systems*, pages 506 – 521, 1996.

[17] Y.-K. Kwok and I. Ahmad. Static Scheduling Algorithms for Allocating Task Graphs to Multiprocessors. *ACM Computing Surveys*, 31(4):406–471, jan 1999.

[18] J.K. Lenstra and A.H.G. Rinnooy Kan. Complexity of scheduling under precedence constraints. *Operations Research*, 26(1):22–35, Jan-Feb 1978.

[19] A. Mandal, K. Kennedy, C. Koelbel, G. Marin, J. Mellor-Crummey, B. Liu, and L. Johnsson. Scheduling strategies for mapping application workflows onto the grid. In *Proceedings: 14th IEEE International HPDC'05*, pages 125–134, NC, Triangle Park, USA, jul 2005.

[20] A. Schaaff, F.L. Petit, E. Prugniel, P.and Slezak, and C. Surace. Workflow in astronomy : the vo france workflow working group experience. In *Astronomical Data Analysis Software and Systems XVII*, 2008.

[21] Y. Tanaka, H. Takemiya, H. Nakada, and S. Sekiguchi. Design, implementation and performance evaluation of gridrpc programming middleware for a large-scale computational grid. In *Proceedings of the Fifth IEEE/ACM Int. Workshop on Grid Computing (GRID'04)*, pages 298–305, Washington DC, USA, 2004.

[22] I.-J. Taylor, E. Deelman, D.-B. Gannon, and M. Shields (Eds.). *Workflows for e-Science*. Springer Verlag, 2007.

[23] H. Topcuouglu, S. Hariri, and M. Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. In *IEEE Trans. on Parallel and Distributed Systems*, pages 260 – 274, 2002.

[24] N. Vydyanathan, U.-V. Catalyurek, T.-M. Kurc, P. Sadayappan, and J.-H. Saltz. Toward optimizing latency under throughput constraints for application workflows on clusters. In Springer, editor, *Euro-Par 2007 Parallel Processing*, volume 4641 of *Lecture Notes in Computer Science*, pages 173–183, 2007.

[25] H. Zhao and R. Sakellariou. Scheduling multiple DAGs onto heterogeneous systems. In *HCW06*, Rhodes, Greece, 2006.

Laboratoire d'Informatique de l'université de Franche-Comté
UFR Sciences et Techniques, 16, route de Gray - 25030 Besançon Cedex (France)

LIFC - Antenne de Belfort : IUT Belfort-Montbéliard, rue Engel Gros, BP 527 - 90016 Belfort Cedex (France)
LIFC - Antenne de Montbéliard : UFR STGI, Pôle universitaire du Pays de Montbéliard - 25200 Montbéliard Cedex (France)

http://lifc.univ-fcomte.fr